Date⇒ 23-02-2021
Module⇒ Backend
Lecture By⇒ Akash Handa
Subject ⇒Postgres

IN PREVIOUS LECTURE **(QUICK RECAP) Date-22/02/2021**	**In Today's Lecture (Overview)**
What is Oltp? Postgres Commands Select PostgreSQL SELECT statement syntax Primary Key Insert into WHERE	Values In Postgres Delete In post Group By

Values In Postgres

VALUES provides a way to generate a "constant table" that can be used in a query without having to actually create and populate a table on-disk. The syntax is
`VALUES (1, 'one'), (2, 'two'), (3, 'three');`
will return a table of two columns and three rows. It's effectively equivalent to:

```
SELECT 1 AS column1, 'one' AS column2
UNION ALL
SELECT 2, 'two'
UNION ALL
SELECT 3, 'three';
```

By default, PostgreSQL assigns the names column1, column2, etc. to the columns of a VALUES table. The column names are not specified by the SQL standard and different database systems do it differently, so it's usually better to override the default names with a table alias list, like this:

```
=> SELECT * FROM (VALUES (1, 'one'), (2, 'two'), (3, 'three')) AS t
(num,letter);
 num | letter
-----+--------
   1 | one
   2 | two
   3 | three
(3 rows)
```

Syntactically, VALUES followed by expression lists is treated as equivalent to:

`SELECT` ***`select_list`*** `FROM` ***`table_expression`***
and can appear anywhere a SELECT can. For example, you can use it as part of a UNION, or attach a ***sort_specification*** (ORDER BY, LIMIT, and/or OFFSET) to it. VALUES is most commonly used as the data source in an INSERT command, and next most commonly as a subquery.

For more information see VALUES.

Delete In post

Summary: in this tutorial, you will learn how to use the **PostgreSQL DELETE** statement to delete data from a table.

Introduction to PostgreSQL DELETE statement

The PostgreSQL `DELETE` statement allows you to delete one or more rows from a table.

The following shows basic syntax of the `DELETE` statement:

```
DELETE FROM table_name
WHERE condition;
```

In this syntax:

> First, specify the name of the table from which you want to delete data after the `DELETE FROM` keywords.
> Second, use a condition in the `WHERE` clause to specify which rows from the table to delete.

The `WHERE` clause is optional. If you omit the `WHERE` clause, the `DELETE` statement will delete all rows in the table.

The `DELETE` statement returns the number of rows deleted. It returns zero if the `DELETE` statement did not delete any row.

To return the deleted row(s) to the client, you use the `RETURNING` clause as follows:

```
DELETE FROM table_name
WHERE condition
RETURNING (select_list | *)
```

```
Code language: SQL (Structured Query Language) (sql)
```

The asterisk (`*`) allows you to return all columns of the deleted row from the `table_name`.

To return specific columns, you specify them after the `RETURNING` keyword.

> Note that the `DELETE` statement only removes data from a table. It doesn't modify the structure of the table. If you want to change the structure of a table such as removing a column, you should use the `ALTER TABLE` statement.

PostgreSQL DELETE statement examples

Let's set up a sample table for the demonstration.

The following statements create a new table called `links` and insert some sample data:

```
DROP TABLE IF EXISTS links;

CREATE TABLE links (
    id serial PRIMARY KEY,
    url varchar(255) NOT NULL,
    name varchar(255) NOT NULL,
    description varchar(255),
    rel varchar(10),
    last_update date DEFAULT now()
);

INSERT INTO
   links
VALUES
   ('1', 'https://www.postgresqltutorial.com', 'PostgreSQL Tutorial', 'Learn
PostgreSQL fast and easy', 'follow', '2013-06-02'),
   ('2', 'http://www.oreilly.com', 'O''Reilly Media', 'O''Reilly Media',
'nofollow', '2013-06-02'),
   ('3', 'http://www.google.com', 'Google', 'Google', 'nofollow',
'2013-06-02'),
   ('4', 'http://www.yahoo.com', 'Yahoo', 'Yahoo', 'nofollow', '2013-06-02'),
   ('5', 'http://www.bing.com', 'Bing', 'Bing', 'nofollow', '2013-06-02'),
   ('6', 'http://www.facebook.com', 'Facebook', 'Facebook', 'nofollow',
'2013-06-01'),
   ('7', 'https://www.tumblr.com/', 'Tumblr', 'Tumblr', 'nofollow',
'2013-06-02'),
   ('8', 'http://www.postgresql.org', 'PostgreSQL', 'PostgreSQL', 'nofollow',
'2013-06-02');
```

Here are the contents of the `links` table:

```
SELECT * FROM links;
```

```
Code language: SQL (Structured Query Language) (sql)
```

	id integer	url character varying (255)	name character varying (255)	description character varying (255)	rel character varying (10)	last_update date
1	1	http://www.postgresqltutori...	PostgreSQL Tutorial	Learn PostgreSQL fast and ...	follow	2013-06-02
2	2	http://www.oreilly.com	O'Reilly Media	O'Reilly Media	nofollow	2013-06-02
3	3	http://www.google.com	Google	Google	nofollow	2013-06-02
4	4	http://www.yahoo.com	Yahoo	Yahoo	nofollow	2013-06-02
5	5	http://www.bing.com	Bing	Bing	nofollow	2013-06-02
6	6	http://www.facebook.com	Facebook	Facebook	nofollow	2013-06-01
7	7	https://www.tumblr.com/	Tumblr	Tumblr	nofollow	2013-06-02
8	8	http://www.postgresql.org	PostgreSQL	PostgreSQL	nofollow	2013-06-02

1) Using PostgreSQL DELETE to delete one row from the table

The following statement uses the `DELETE` statement to delete one row with the id 8 from the `links` table:

```
DELETE FROM links
WHERE id = 8;
Code language: SQL (Structured Query Language) (sql)
```

The statement returns 1 indicated that one row has been deleted:

```
DELETE 1
Code language: Shell Session (shell)
```

The following statement uses the `DELETE` statement to delete the row with id 10:

```
DELETE FROM links
WHERE id = 10;
Code language: SQL (Structured Query Language) (sql)
```

Since the row with id 10 does not exist, the statement returns 0:

```
DELETE 0
Code language: Shell Session (shell)
```

2) Using PostgreSQL DELETE to delete a row and return the deleted row

The following statement deletes the row with id 7 and returns the deleted row to the client:

```
DELETE FROM links
WHERE id = 7
RETURNING *;
Code language: SQL (Structured Query Language) (sql)
```

PostgreSQL returns the following deleted row:

	id integer	url character varying (255)	name character varying (255)	description character varying (255)	rel character varying (10)	last_update date
1	7	https://www.tumblr.com/	Tumblr	Tumblr	nofollow	2013-06-02

3) Using PostgreSQL DELETE to delete multiple rows from the table

The following statement deletes two rows from the `links` table and return the values in the `id` column of deleted rows:

```
DELETE FROM links
WHERE id IN (6,5)
RETURNING *;
```

Code language: SQL (Structured Query Language) (sql)

Output:

4) Using PostgreSQL DELETE to delete all rows from the table

The following statement uses the `DELETE` statement without a `WHERE` clause to delete all rows from the `links` table:

```
DELETE FROM links;
```

Group By

The `GROUP BY` clause divides the rows returned from the `SELECT` statement into groups. For each group, you can apply an aggregate function e.g., `SUM()` to calculate the sum of items or `COUNT()` to get the number of items in the groups.

The following statement illustrates the basic syntax of the `GROUP BY` clause:

```
SELECT
   column_1,
   column_2,
   ...,
   aggregate_function(column_3)
FROM
   table_name
GROUP BY
   column_1,
   column_2,
   ...;
```

Code language: SQL (Structured Query Language) (sql)

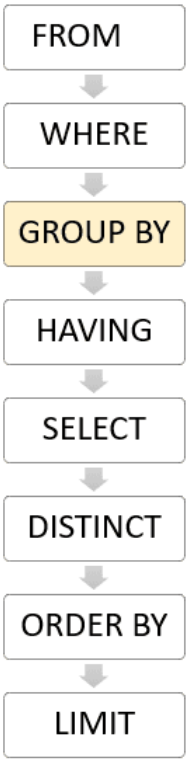
In this syntax:

First, select the columns that you want to group e.g., `column1` and `column2`, and column that you want to apply an aggregate function (`column3`).
Second, list the columns that you want to group in the `GROUP BY` clause.

The statement clause divides the rows by the values of the columns specified in the `GROUP BY` clause and calculates a value for each group.

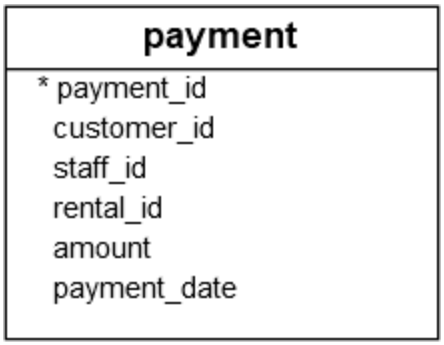
It's possible to use other clauses of the `SELECT` statement with the `GROUP BY` clause.

PostgreSQL evaluates the `GROUP BY` clause after the `FROM` and `WHERE` clauses and before the `HAVING` `SELECT`, `DISTINCT`, `ORDER BY` and `LIMIT` clauses.



PostgreSQL GROUP BY clause examples

Let's take a look at the `payment` table in the sample database.



1) Using PostgreSQL `GROUP BY` without an aggregate function example

You can use the `GROUP BY` clause without applying an aggregate function. The following query gets data from the `payment` table and groups the result by customer id.

```
SELECT
    customer_id
FROM
    payment
GROUP BY
    customer_id;
```

Code language: SQL (Structured Query Language) (sql)

	customer_id smallint
1	184
2	87
3	477
4	273
5	550
6	51
7	394
8	272
9	70

In this case, the `GROUP BY` works like the `DISTINCT` clause that removes duplicate rows from the result set.

2) Using PostgreSQL `GROUP BY` with `SUM()` function example

The `GROUP BY` clause is useful when it is used in conjunction with an aggregate function.

For example, to select the total amount that each customer has been paid, you use the `GROUP BY` clause to divide the rows in the `payment` table into groups grouped by customer id. For each group, you calculate the total amounts using the `SUM()` function.

The following query uses the `GROUP BY` clause to get total amount that each customer has been paid:

```
SELECT
	customer_id,
	SUM (amount)
FROM
	payment
GROUP BY
	customer_id;
```

Code language: SQL (Structured Query Language) (sql)

	customer_id smallint	sum numeric
1	184	80.80
2	87	137.72
3	477	106.79
4	273	130.72
5	550	151.69
6	51	123.70
7	394	77.80
8	272	65.87
9	70	75.83
10	190	102.75
11	350	63.79

The `GROUP BY` clause sorts the result set by customer id and adds up the amount that belongs to the same customer. Whenever the `customer_id` changes, it adds the row to the returned result set.